1

Hello, Android

COPYRIGHTED MATERIAL

WHAT'S IN THIS CHAPTER?

- A background to mobile application development
- What Android is (and what it isn't)
- An introduction to the Android SDK features
- What devices Android runs on
- Why develop for mobile and Android?
- An introduction to the SDK and the Android development framework

Whether you're an experienced mobile engineer, a desktop or web developer, or a complete programming novice, Android represents an exciting new opportunity to write innovative applications for mobile devices.

Despite the name, Android will not help you create an unstoppable army of emotionless robot warriors on a relentless quest to cleanse the earth of the scourge of humanity. Instead, Android is an open-source software stack that includes the operating system, middleware, and key mobile applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets.

Small, stylish, and versatile, modern mobile devices have become powerful tools that incorporate cameras, media players, GPS systems, and touchscreens. As technology has evolved, mobile phones have become about more than simply making calls, but their software and development platforms have struggled to keep pace.

Until recently, mobile phones were largely closed environments built on highly fragmented, proprietary operating systems that required proprietary development tools. The phones themselves often prioritized native applications over those written by third parties. This has introduced an artificial barrier for developers hoping to build on increasingly powerful mobile hardware.

In Android, native and third-party applications are written with the same APIs and executed on the same run time. These APIs feature hardware sensor access, video recording, location-based services, support for background services, map-based activities, relational databases, inter-application communication, and 2D and 3D graphics.

Using this book, you will learn how to use these APIs to create your own Android applications. In this chapter you'll learn some mobile development guidelines and be introduced to the features available from the Android development platform.

Android has powerful APIs, excellent documentation, a thriving developer community, and no development or distribution costs. As mobile devices continue to increase in popularity, this is an exciting opportunity to create innovative mobile phone applications no matter what your development experience.

A LITTLE BACKGROUND

In the days before Twitter and Facebook, when Google was still a twinkle in its founders' eyes and dinosaurs roamed the earth, mobile phones were just that — portable phones small enough to fit inside a briefcase, featuring batteries that could last up to several hours. They did however offer the freedom to make calls without being physically connected to a landline.

Increasingly small, stylish, and powerful mobile phones are now as ubiquitous as they are indispensable. Hardware advancements have made mobiles smaller and more efficient while including an increasing number of peripherals.

After first getting cameras and media players, mobiles now include GPS systems, accelerometers, and touch screens. While these hardware innovations should prove fertile ground for software development, the applications available for mobile phones have generally lagged behind the hardware.

The Not-So-Distant Past

Historically, developers, generally coding in low-level C or C++, have needed to understand the specific hardware they were coding for, generally a single device or possibly a range of devices from a single manufacturer. As hardware technology and mobile Internet access has advanced, this closed approach has become outmoded.

More recently, platforms like Symbian have been created to provide developers with a wider target audience. These systems have proven more successful in encouraging mobile developers to provide rich applications that better leverage the hardware available.

These platforms offer some access to the device hardware, but require the developer to write complex C/C++ code and make heavy use of proprietary APIs that are notoriously difficult to work with. This difficulty is amplified for applications that must work on different hardware implementations and those that make use of a particular hardware feature, like GPS.

In more recent years, the biggest advance in mobile phone development was the introduction of Java-hosted MIDlets. MIDlets are executed on a Java virtual machine, a process that abstracts the underlying hardware and lets developers create applications that run on the wide variety of devices that supports the Java run time. Unfortunately, this convenience comes at the price of restricted access to the device hardware.

In mobile development it was considered normal for third-party applications to receive different hardware access and execution rights from those given to native applications written by the phone manufacturers, with MIDlets often receiving few of either.

The introduction of Java MIDlets expanded developers' audiences, but the lack of low-level hardware access and sandboxed execution meant that most mobile applications are regular desktop programs or web sites designed to render on a smaller screen, and do not take advantage of the inherent mobility of the handheld platform.

The Future

Android sits alongside a new wave of mobile operating systems designed for increasingly powerful mobile hardware. Windows Mobile, the Apple iPhone, and the Palm Pre now provide a richer, simplified development environment for mobile applications. However, unlike Android, they're built on proprietary operating systems that in some cases prioritize native applications over those created by third parties, restrict communication among applications and native phone data, and restrict or control the distribution of third-party apps to their platforms.

Android offers new possibilities for mobile applications by offering an open development environment built on an open-source Linux kernel. Hardware access is available to all applications through a series of API libraries, and application interaction, while carefully controlled, is fully supported.

In Android, all applications have equal standing. Third-party and native Android applications are written with the same APIs and are executed on the same run time. Users can remove and replace any native application with a third-party developer alternative; even the dialer and home screens can be replaced.

WHAT IT ISN'T

As a disruptive addition to a mature field, it's not hard to see why there has been some confusion about what exactly Android is. Android is **not:**

- **A Java ME implementation** Android applications are written in the Java language, but they are not run within a Java ME virtual machine, and Java-compiled classes and executables will not run natively in Android.
- **Part of the Linux Phone Standards Forum (LiPS) or the Open Mobile Alliance (OMA)** Android runs on an open-source Linux kernel, but, while their goals are similar, Android's complete software stack approach goes further than the focus of these standards-defining organizations.
- **Simply an application layer (like UIQ or S60)** While Android does include an application layer, ''Android'' also describes the entire software stack encompassing the underlying operating system, the API libraries, and the applications themselves.
- **A mobile phone handset** Android includes a reference design for mobile handset manufacturers, but there is no single ''Android phone.'' Instead, Android has been designed to support many alternative hardware devices.
- **Google's answer to the iPhone** The iPhone is a fully proprietary hardware and software platform released by a single company (Apple), while Android is an open-source software

stack produced and supported by the Open Handset Alliance and designed to operate on any handset that meets the requirements. Google has now released its first direct-to-consumer handset, the Nexus 1, but this device remains simply one hardware implementation running on the Android platform.

ANDROID: AN OPEN PLATFORM FOR MOBILE DEVELOPMENT

Google's Andy Rubin describes Android as:

> *The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.* (`http://googleblog.blogspot.com/2007/11/wheres-my-gphone.html`)

Put simply, Android is a combination of three components:

- A free, open-source operating system for mobile devices
- An open-source development platform for creating mobile applications
- Devices, particularly mobile phones, that run the Android operating system and the applications created for it

More specifically, Android is made up of several necessary and dependent parts, including the following:

- A hardware reference design that describes the capabilities required for a mobile device to support the software stack.
- A Linux operating system kernel that provides low-level interface with the hardware, memory management, and process control, all optimized for mobile devices.
- Open-source libraries for application development, including SQLite, WebKit, OpenGL, and a media manager.
- A run time used to execute and host Android applications, including the Dalvik virtual machine and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on mobile devices.
- An application framework that agnostically exposes system services to the application layer, including the window manager and location manager, content providers, telephony, and sensors.
- A user interface framework used to host and launch applications.
- Preinstalled applications shipped as part of the stack.
- A software development kit used to create applications, including tools, plug-ins, and documentation.

What really makes Android compelling is its open philosophy, which ensures that you can fix any deficiencies in user interface or native application design by writing an extension or replacement. Android

provides you, as a developer, with the opportunity to create mobile phone interfaces and applications designed to look, feel, and function exactly as you imagine them.

NATIVE ANDROID APPLICATIONS

Android phones will normally come with a suite of generic preinstalled applications that are part of the Android Open Source Project (AOSP), including, but not necessarily limited to:

- An e-mail client
- An SMS management application
- A full PIM (personal information management) suite including a calendar and contacts list
- A WebKit-based web browser
- A music player and picture gallery
- A camera and video recording application
- A calculator
- The home screen
- An alarm clock

In many cases Android devices will also ship with the following proprietary Google mobile applications:

- The Android Market client for downloading third-party Android applications
- A fully-featured mobile Google Maps application including StreetView, driving directions and turn-by-turn navigation, satellite view, and traffic conditions
- The Gmail mail client
- The Google Talk instant-messaging client
- The YouTube video player

The data stored and used by many of these native applications — like contact details — are also available to third-party applications. Similarly, your applications can handle events such as incoming calls or new SMS messages.

The exact makeup of the applications available on new Android phones is likely to vary based on the hardware manufacturer and/or the phone carrier or distributor.

The open-source nature of Android means that carriers and OEMs can customize the user interface and the applications bundled with each Android device. Several OEMs have done this, including HTC with the Sense UI, Motorola with MotoBlur, and Sony Ericsson's custom UI.

It's important to note that for compatible devices, the underlying platform and SDK remain consistent across OEM and carrier variations. The look and feel of the user interface may vary, but your applications will function in the same way across all compatible Android devices.

ANDROID SDK FEATURES

The true appeal of Android as a development environment lies in the APIs it provides.

As an application-neutral platform, Android gives you the opportunity to create applications that are as much a part of the phone as anything provided out of the box. The following list highlights some of the most noteworthy Android features:

- No licensing, distribution, or development fees or release approval processes
- Wi-Fi hardware access
- GSM, EDGE, and 3G networks for telephony or data transfer, enabling you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks
- Comprehensive APIs for location-based services such as GPS
- Full multimedia hardware control, including playback and recording with the camera and microphone
- APIs for using sensor hardware, including accelerometers and the compass
- Libraries for using Bluetooth for peer-to-peer data transfer
- IPC message passing
- Shared data stores
- Background applications and processes
- Home-screen Widgets, Live Folders, and Live Wallpaper
- The ability to integrate application search results into the system search
- An integrated open-source HTML5 WebKit-based browser
- Full support for applications that integrate map controls as part of their user interface
- Mobile-optimized hardware-accelerated graphics, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0
- Media libraries for playing and recording a variety of audio/video or still image formats
- Localization through a dynamic resource framework
- An application framework that encourages reuse of application components and the replacement of native applications

Access to Hardware, Including Camera, GPS, and Accelerometer

Android includes API libraries to simplify development involving the device hardware. These ensure that you don't need to create specific implementations of your software for different devices, so you can create Android applications that work as expected on any device that supports the Android software stack.

The Android SDK includes APIs for location-based hardware (such as GPS), the camera, audio, network connections, Wi-Fi, Bluetooth, accelerometers, the touchscreen, and power management. You can explore the possibilities of some of Android's hardware APIs in more detail in Chapters 11 through 14.

Native Google Maps, Geocoding, and Location-Based Services

Native map support lets you create a range of map-based applications that leverage the mobility of Android devices. Android lets you create activities that include interactive Google Maps as part of your user interface, with full access to maps that you can control programmatically and annotate using Android's rich graphics library.

Android's location-based services manage technologies like GPS and Google's GSM cell-based location technology to determine the device's current position. These services enforce an abstraction from specific location-detecting technology and let you specify minimum requirements (e.g., accuracy or cost) rather than choosing a particular technology. They also mean that your location-based applications will work no matter what technology the host handset supports.

To combine maps with locations, Android includes an API for forward and reverse geocoding that lets you find map coordinates for an address, and the address of a map position.

You'll learn the details of using maps, the Geocoder, and location-based services in Chapter 8.

Background Services

Android supports applications and services designed to run invisibly in the background.

Modern mobiles are by nature multifunction devices; however, their limited screen sizes means that generally only one interactive application can be visible at any time. Platforms that don't support background execution limit the viability of applications that don't need your constant attention.

Background services make it possible to create invisible application components that perform automatic processing without direct user action. Background execution allows your applications to become event-driven and to support regular updates, which is perfect for monitoring game scores or market prices, generating location-based alerts, or prioritizing and prescreening incoming calls and SMS messages.

Learn more about how to get the most out of background services in Chapter 9.

SQLite Database for Data Storage and Retrieval

Rapid and efficient data storage and retrieval are essential for a device whose storage capacity is limited by its compact nature.

Android provides a lightweight relational database for each application using SQLite. Your applications can take advantage of this managed relational database engine to store data securely and efficiently.

By default each application database is *sandboxed* — its content is available only to the application that created it — but Content Providers supply a mechanism for the managed sharing of these application databases.

Databases and Content Providers are covered in detail in Chapter 7.

Shared Data and Interapplication Communication

Android includes three techniques for transmitting information from your applications for use elsewhere: Notifications, Intents, and Content Providers.

Notifications are the standard means by which a mobile device traditionally alerts users. Using the API you can trigger audible alerts, cause vibration, and flash the device's LED, as well as control status bar notification icons, as shown in Chapter 9.

Intents provide a mechanism for message-passing within and between applications. Using Intents you can broadcast a desired action (such as dialing the phone or editing a contact) system-wide for other applications to handle. Intents are an important core component of Android and are covered in depth in Chapter 5.

Finally, you can use *Content Providers* to give managed access to your application's private databases. The data stores for native applications, such as the contact manager, are exposed as Content Providers so you can create your own applications that read or modify this data. Chapter 7 covers Content Providers in detail, including the native providers, and demonstrates how to create and use providers of your own.

Using Widgets, Live Folders, and Live Wallpaper to Enhance the Home Screen

Widgets, Live Folders, and Live Wallpaper let you create dynamic application components that provide a window into your applications or offer useful and timely information directly on the home screen.

If you offer a way for users to interact with your application directly from the home screen, they get instant access to interesting information without needing to open an application, and you get a dynamic shortcut into your application.

You'll learn how to create application components for the home screen in Chapter 10.

Extensive Media Support and 2D/3D Graphics

Bigger screens and brighter, higher-resolution displays have helped make mobiles multimedia devices. To help you make the most of the hardware available, Android provides graphics libraries for 2D canvas drawing and 3D graphics with OpenGL.

Android also offers comprehensive libraries for handling still images, video, and audio files, including the MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, and GIF formats.

2D and 3D graphics are covered in depth in Chapter 15, while Android media management libraries are covered in Chapter 11.

Optimized Memory and Process Management

Android's process and memory management is a little unusual. Like Java and .NET, Android uses its own run time and virtual machine to manage application memory. Unlike with either of these other frameworks, the Android run time also manages the process lifetimes. Android ensures application responsiveness by stopping and killing processes as necessary to free resources for higher-priority applications.

In this context, the highest priority is given to the application with which the user is interacting. Ensuring that your applications are prepared for a swift death but are still able to remain responsive, and to

update or restart in the background if necessary, is an important consideration in an environment that does not allow applications to control their own lifetimes.

You will learn more about the Android application life cycle in Chapter 3.

INTRODUCING THE OPEN HANDSET ALLIANCE

The Open Handset Alliance (OHA) is a collection of more than 50 technology companies, including hardware manufacturers, mobile carriers, and software developers. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm. In their own words, the OHA represents the following:

> *A commitment to openness, a shared vision for the future, and concrete plans to make the vision a reality. To accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. (*`http://www.openhandsetalliance.com/`*)*

The OHA hopes to deliver a better mobile software experience for consumers by providing the platform needed for innovative mobile development at a faster rate and with higher quality than existing platforms, without licensing fees for either software developers or handset manufacturers.

WHAT DOES ANDROID RUN ON?

The first Android mobile handset, the T-Mobile G1, was released in the United States in October 2008. By the end of 2009 over 20 Android-compatible handsets had been launched or announced in more than 26 countries on 32 different carrier networks.

Rather than being a mobile OS created for a single hardware implementation, Android is designed to support a large variety of hardware platforms, from WVGA phones with hard keyboards to QVGA devices with resistive touchscreens.

Beyond that, with no licensing fees or proprietary software, the cost to handset manufacturers for providing Android handsets, and potentially other Android-powered devices, is comparatively low. Many people now expect that the advantages of Android as a platform for creating powerful applications will encourage device manufacturers to produce increasingly tailored hardware.

WHY DEVELOP FOR MOBILE?

In market terms, the emergence of modern mobile smartphones and *superphones* — multifunction devices including a phone but featuring a full-featured web browser, cameras, media players, Wi-Fi, and location-based services — has fundamentally changed the way people interact with their mobile devices and access the Internet. Mobile-phone ownership easily surpasses computer ownership in many countries; 2009 marked the year that more people accessed the Internet for the first time from a mobile phone rather than a PC.

The increasing popularity of modern smartphones, combined with the increasing availability of flat-rate, affordable data plans and Wi-Fi, has created a growth market for advanced mobile applications.

The ubiquity of mobile phones, and our attachment to them, makes them a fundamentally different platform for development from PCs. With a microphone, a camera, a touchscreen, location detection, and environmental sensors, a phone can effectively become an extension of your own perceptions.

With the average Android user installing and using around 40 apps, mobile applications have changed the way people use their phones. This gives you, the application developer, a unique opportunity to create dynamic, compelling new applications that become a vital part of people's lives.

WHY DEVELOP FOR ANDROID?

If you have a background in mobile application development, you don't need me to tell you that:

- A lot of what you can do with Android is already possible.
- But doing it is painful.

Android represents a clean break, a mobile framework based on the reality of modern mobile devices designed by developers, for developers.

With a simple and powerful SDK, no licensing fees, excellent documentation, and a thriving developer community, Android represents an excellent opportunity to create software that changes how and why people use their mobile phones.

From a commercial perspective Android:

- Requires no certification for becoming an Android developer
- Provides the Android Market for distribution and monetization of your applications
- Has no approval process for application distribution
- Gives you total control over your brand and access to the user's home screen

What Has and Will Continue to Drive Android Adoption?

Android is targeted primarily at developers, with Google and the OHA betting that the way to deliver better mobile software to consumers is to make it easier for developers to write it themselves.

As a development platform, Android is powerful and intuitive, letting developers who have never programmed for mobile devices create useful applications quickly and easily. It's easy to see how innovative Android applications could create demand for the devices necessary to run them, particularly if developers write applications for Android because they *can't* write them for other platforms.

Open access to the nuts and bolts of the underlying system is what's always driven software development and platform adoption. The Internet's inherent openness and neutrality have seen it become the platform for a multibillion-dollar industry within 10 years of its inception. Before that, it was open systems like Linux and the powerful APIs provided as part of the Windows operating system that enabled the explosion in personal computers and the movement of computer programming from the arcane to the mainstream.

This openness and power ensure that anyone with the inclination can bring a vision to life at minimal cost.

What Does It Have That Others Don't?

Many of the features listed previously, such as 3D graphics and native database support, are also available in other mobile SDKs. Here are some of the unique features that set Android apart:

- **Google Map applications** Google Maps for Mobile has been hugely popular, and Android offers a Google Map as an atomic, reusable control for use in your applications. The Map View lets you display, manipulate, and annotate a Google Map within your Activities to build map-based applications using the familiar Google Maps interface.
- **Background services and applications** Background services let you create an application that uses an event-driven model, working silently while other applications are being used or while your mobile sits ignored until it rings, flashes, or vibrates to get your attention. Maybe it's a streaming music player, an application that tracks the stock market, alerting you to significant changes in your portfolio, or a service that changes your ringtone or volume depending on your current location, the time of day, and the identity of the caller.
- **Shared data and interprocess communication** Using Intents and Content Providers, Android lets your applications exchange messages, perform processing, and share data. You can also use these mechanisms to leverage the data and functionality provided by the native Android applications. To mitigate the risks of such an open strategy, each application's process, data storage, and files are private unless explicitly shared with other applications via a full permission-based security mechanism detailed in Chapter 15.
- **All applications are created equal** Android doesn't differentiate between native applications and those developed by third parties. This gives consumers unprecedented power to change the look and feel of their devices by letting them completely replace every native application with a third-party alternative that has access to the same underlying data and hardware.
- **Home-screen Widgets, Live Folders, Live Wallpaper, and the quick search box** Using Widgets, Live Folders, and Live Wallpaper, you can create windows into your application from the phone's home screen. The quick search box lets you integrate search results from your application directly into the phone's search functionality.

Changing the Mobile Development Landscape

Existing mobile development platforms have created an aura of exclusivity around mobile development. Whether by design or as a side effect of the cost, complexity, or necessity for approval involved in developing native applications, many mobile phones remain almost exactly as they were when first purchased.

In contrast, Android allows, even encourages, radical change. As consumer devices, Android handsets ship with a core set of the standard applications that consumers demand on a new phone, but the real power lies in users' ability to completely change how their devices look, feel, and function.

Android gives developers a great opportunity. All Android applications are a native part of the phone, not just software that's run in a sandbox on top of it. Rather than writing small-screen versions of

software that can be run on low-power devices, you can now write mobile applications that change the way people use their phones.

While Android will still have to compete with existing and future mobile development platforms as an open-source developer framework, the strength of the development kit is very much in its favor. Certainly its free and open approach to mobile application development, with total access to the phone's resources, is a giant step in the right direction.

INTRODUCING THE DEVELOPMENT FRAMEWORK

With the PR job done, it's time to look at how you can start developing applications for Android. Android applications are written with Java as a programming language but executed by means of a custom virtual machine called Dalvik rather than a traditional Java VM.

Later in this chapter you'll be introduced to the framework, starting with a technical explanation of the Android software stack, a look at what's included in the SDK, an introduction to the Android libraries, and a look at the Dalvik virtual machine.

Each Android application runs in a separate process within its own Dalvik instance, relinquishing all responsibility for memory and process management to the Android run time, which stops and kills processes as necessary to manage resources.

Dalvik and the Android run time sit on top of a Linux kernel that handles low-level hardware interaction, including drivers and memory management, while a set of APIs provides access to all the underlying services, features, and hardware.

What Comes in the Box

The Android software development kit (SDK) includes everything you need to start developing, testing, and debugging Android applications. Included in the SDK download are:

- **The Android APIs** The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries used at Google to create native Android applications.
- **Development tools** So you can turn Android source code into executable Android applications, the SDK includes several development tools that let you compile and debug your applications. You will learn more about the developer tools in Chapter 2.
- **The Android Virtual Device Manager and Emulator** The Android Emulator is a fully interactive Android device emulator featuring several alternative skins. The emulator runs within an Android Virtual Device that simulates the device hardware configuration. Using the emulator you can see how your applications will look and behave on a real Android device. All Android applications run within the Dalvik VM, so the software emulator is an excellent environment — in fact, as it is hardware-neutral, it provides a better independent test environment than any single hardware implementation.
- **Full documentation** The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them. In addition to

the code documentation, Android's reference documentation explains how to get started and gives detailed explanations of the fundamentals behind Android development.

- **Sample code** The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available with Android, as well as simple programs that highlight how to use individual API features.
- **Online support** Android has rapidly generated a vibrant developer community. The Google Groups at `http://developer.android.com/resources/community-groups.html` are active forums of Android developers with regular input from the Android engineering and developer relations teams at Google. StackOverflow at `http://www.stackoverflow.com/questions/tagged/android` has also become a popular destination for Android questions.

For those using the popular Eclipse IDE, Android has released a special plug-in that simplifies project creation and tightly integrates Eclipse with the Android Emulator and debugging tools. The features of the ADT plug-in are covered in more detail in Chapter 2.

Understanding the Android Software Stack

The Android software stack is composed of the elements shown in Figure 1-1 and described in further detail after it. Put simply, a Linux kernel and a collection of C/C++ libraries are exposed through an application framework that provides services for, and management of, the run time and applications.

- **Linux kernel** Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.
- **Libraries** Running on top of the kernel, Android includes various C/C++ core libraries such as libc and SSL, as well as:
  - A media library for playback of audio and video media
  - A surface manager to provide display management
  - Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
  - SQLite for native database support
  - SSL and WebKit for integrated web browser and Internet security
- **Android run time** What makes an Android phone an Android phone rather than a mobile Linux implementation is the Android run time. Including the core libraries and the Dalvik virtual machine, the Android run time is the engine that powers your applications and, along with the libraries, forms the basis for the application framework.
  - **Core libraries** While Android development is done in Java, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries as well as the Android-specific libraries.
  - **Dalvik virtual machine** Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

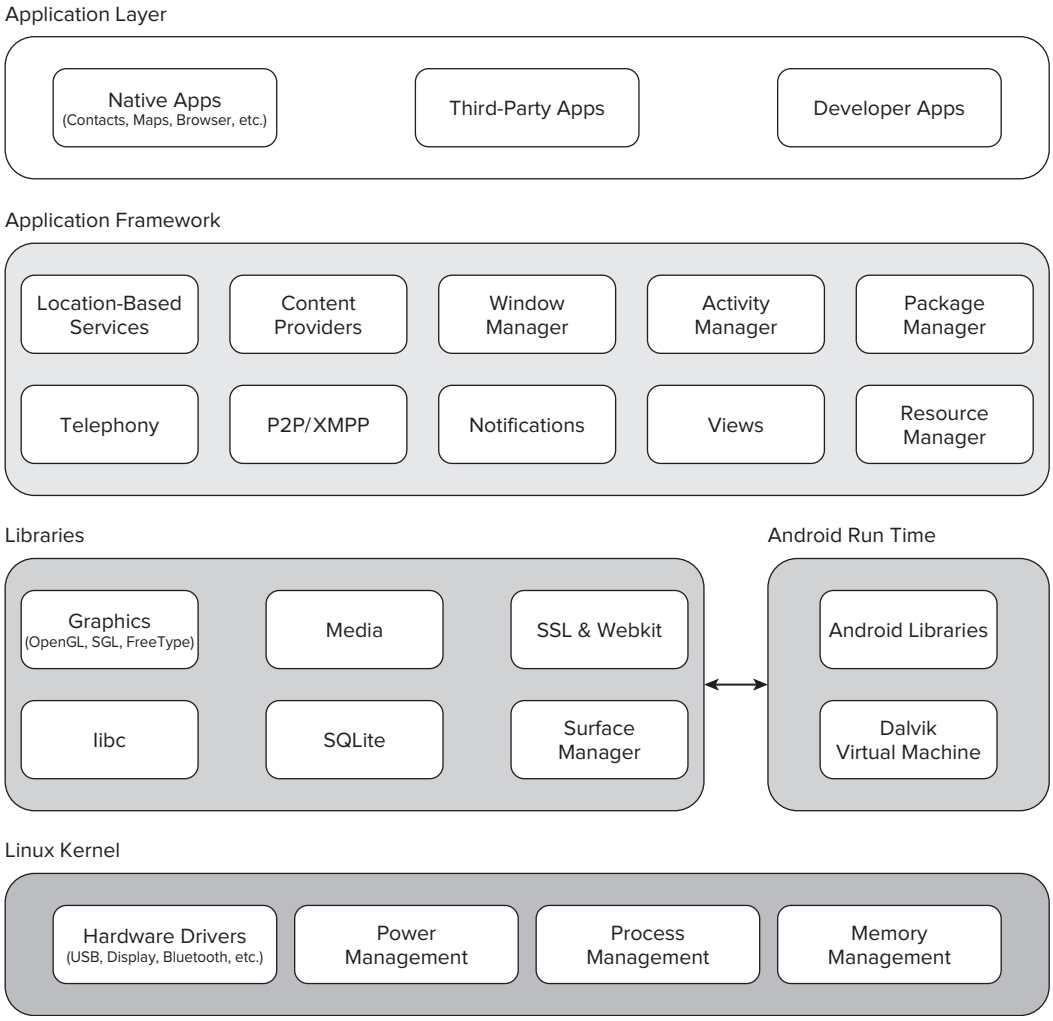

FIGURE 1-1

- **Application framework** The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- **Application layer** All applications, both native and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android run time, using the classes and services made available from the application framework.

The Dalvik Virtual Machine

One of the key elements of Android is the Dalvik virtual machine. Rather than use a traditional Java virtual machine (VM) such as Java ME (Java Mobile Edition), Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.

The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management. It's also possible to write C/C++ applications that run directly on the underlying Linux OS. While you *can* do this, in most cases there's no reason you should need to.

If the speed and efficiency of C/C++ is required for your application, Android now provides a Native Development Kit (NDK). The NDK is designed to enable you to create C++ libraries using the libc and libm libraries, along with native access to OpenGL.

This book focuses exclusively on writing applications that run within Dalvik using the SDK. If your inclinations run toward exploring the Linux kernel and C/C++ underbelly of Android, modifying Dalvik, or otherwise tinkering with things under the hood, check out the Android Internals Google Group at `http://groups.google.com/group/android-internals`

While use of the NDK is encouraged where needed, details of its use have not been included within this book.

All Android hardware and system service access is managed using Dalvik as a middle tier. By using a VM to host application execution, developers have an abstraction layer that ensures they never have to worry about a particular hardware implementation.

The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory footprint. You create `.dex` executables by transforming Java language compiled classes using the tools supplied within the SDK. You'll learn more about how to create Dalvik executables in the next chapter.

Android Application Architecture

Android's architecture encourages the concept of component reuse, enabling you to publish and share Activities, Services, and data with other applications, with access managed by the security restrictions you put in place.

The same mechanism that lets you produce a replacement contact manager or phone dialer can let you expose your application components to let other developers create new UI front ends and functionality extensions, or otherwise build on them.

The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:

- **Activity Manager** Controls the life cycle of your Activities, including management of the Activity stack described in Chapter 3.
- **Views** Used to construct the user interfaces for your Activities, as described in Chapter 4.
- **Notification Manager** Provides a consistent and nonintrusive mechanism for signaling your users, as described in Chapter 9.
- **Content Providers** Let your applications share data, as described in Chapter 7.
- **Resource Manager** Supports non-code resources like strings and graphics to be externalized, as shown in Chapter 3.

Android Libraries

Android offers a number of APIs for developing your applications. Rather than list them all here, I refer you to the documentation at `http://developer.android.com/reference/packages.html`, which gives a complete list of packages included in the Android SDK.

Android is intended to target a wide range of mobile hardware, so be aware that the suitability and implementation of some of the advanced or optional APIs may vary depending on the host device.

SUMMARY

This chapter explained that despite significant advances in the hardware features available on modern mobile phones, the software has lagged. Hard-to-use development kits, hardware-specific APIs, and a lack of openness have stifled innovation in mobile software.

Android offers an opportunity for developers to create innovative software applications for mobile devices without the restrictions generally associated with the existing proprietary mobile development frameworks.

You were shown the complete Android software stack, which includes not only an application layer and development toolkit but also the Dalvik VM, a custom run time, core libraries, and a Linux kernel, all of which are available as open source.

You also learned:

- How handsets with an expanding range of hardware features have created demand for tools that give developers better access to these features.
- About some of the features available to developers using Android, including native map support, hardware access, background services, interprocess messaging, shared databases, and 2D and 3D graphics.
- That all Android applications are built equal, allowing users to completely replace one application, even a core native application, with another.
- That the Android SDK includes developer tools, APIs, and comprehensive documentation.

The next chapter will help you get started by downloading and installing the Android SDK and setting up an Android development environment in Eclipse.

You'll also learn how to use the Android developer tools plug-in to streamline development, testing, and debugging before creating your first Android application.

After learning about the building blocks of Android applications, you'll be introduced to the different types of applications you can create, and you'll start to understand some of the design considerations that should go into developing applications for mobile devices.